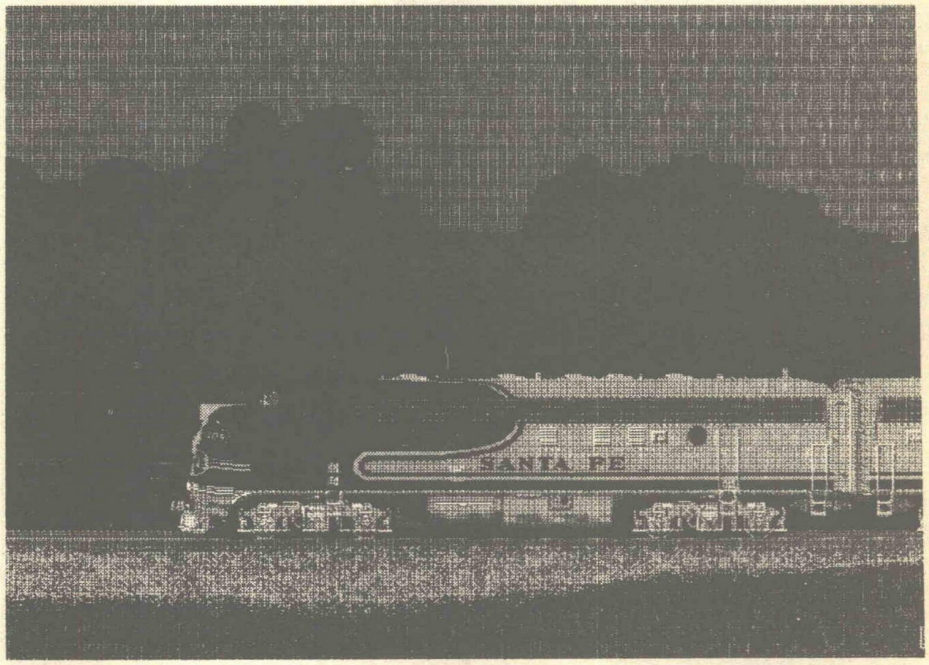
ชูเกียรติ แสงมณี และ ส.อ.บุญเรือน คงเขียว

SOFT MAIL

เทคนิควิธีการนำฟอนต์ CU Writer และภาพ .PCX มาแสดงบนกราฟิกสี

มุ่งเข้ามาสู่เรื่องของการทำตัวอักษรพิเศษและการนำภาพมาแสดงบนกราฟิกสี



มือเร็ว ๆ นี้ได้มีข่าวครึกโครมเกี่ยวกับภาพอนาจารที่ออกมาสู่วงการคอมพิวเตอร์ซอฟต์แวร์ ทั้งผู้ปกครองและสื่อมวลชนต่างก็เป็นห่วงเรื่องนี้มาก โดยเฉพาะเด็ก ๆ หรือเยาวชนของชาติที่อาจจะลุ่มหลงหมกมุ่นกับภาพเหล่านั้น ความจริงแล้วภาพพวกนี้ไม่ใช่ของใหม่สำหรับวงการคอมพิวเตอร์เลย เราพบเห็นกันมานานแล้วตั้งแต่เราใช้คอมพิวเตอร์กราฟิก ตอนนั้นเครื่องคอมพิวเตอร์ยังมีราคาค่อนข้างสูง และซอฟต์แวร์ต่าง ๆ ก็ยังอยู่ในมือของผู้ใหญ่ ปัจจุบันเครื่องคอมพิวเตอร์มีอยู่ตามบ้านมากขึ้น ซอฟต์แวร์ก็มีราคาถูกและหาได้ง่ายมาก การที่เรจะมาเรียกร้องให้กวาดล้างหรือทำลายให้สิ้นซากนั้นเป็นเรื่องที่เป็นไปไม่ได้ เพราะซอฟต์แวร์คอมพิวเตอร์ไม่สามารถจะควบคุมหรือตรวจสอบได้เหมือนกับแถบแม่เหล็กชนิดอื่น ๆ

ในความเป็นจริงเราต้องยอมรับว่าเทคโนโลยีของการสร้างอาวุธสงครามและโครงการอวกาศต่าง ๆ ได้ส่งผลให้เกิดการพัฒนาเทคโนโลยีใหม่ให้เราได้ใช้กันทุกวันนี้ เช่นเดียวกัน ซอฟต์แวร์เหล่านี้ก็มีส่วนในการพัฒนาซอฟต์แวร์อื่น ๆ เช่นกัน เพียงแต่เรานำไปใช้ในทางที่ไม่ถูกไม่ควร แต่ถ้าเราทุกคนมีจริยธรรม มีความรับผิดชอบต่อสังคมต่อส่วนรวม ช่วยกันดูแลและยับยั้งไม่ให้เผยแพร่ไปสู่เยาวชนของชาติ ก็อาจจะเป็นอีกหนทางหนึ่งที่เราพอจะช่วยกันได้

มาถึงตอนนี้ผู้เขียนก็ต้องขอหลีกเลี่ยงการเสนอภาพกราฟิกในฟอร์แมต GIF ที่มีความนิยมมากในเรื่องโป๊ ๆ เปลือย ๆ แม้ว่าภาพที่ดี ๆ ก็มี แต่ผู้เขียนก็ไม่อยากเสี่ยงที่จะนำมาเสนอ

มาเข้าเรื่องของเราเลยดีกว่า จาก 2 ตอนที่แล้วเราได้ศึกษาถึงการทำงานแบบกราฟิกมาพอสมควรแล้ว ตอนนี้เราจะนำเอาความรู้เดิมมาใช้งานจริง ๆ ดูบ้าง เริ่มจากโปรแกรมยอดนิยมประเภท WYSIWYG (What You See Is What You Get) ของคนไทยมาเป็นแบบอย่าง (ผู้เขียนว่าเป็นแบบอย่างจริง ๆ ไม่ได้ลอกเขามาอย่างที่ชาวเรานิยมกัน) มีคนเป็นจำนวนมากที่ต้องใช้งานการพิมพ์เอกสาร หรือที่ภาษาฝรั่งเรียกว่า Word Processor คงหนีไม่พ้น CU Writer และ PC Word ของราชวิถี แต่เรารู้จัก CU Writer ในแบบ WYSIWYG มาก่อน จึงต้องขอยกมาเป็นแบบอย่างในการศึกษาของเรา และเพื่อความสะดวกเราก็ขอยืมรูปแบบตัวหนังสือ (Font)

ทั้ง NORMAL.FON และ ITALIC.FON มาใช้งานเลย แต่ถ้าใครมีเวลาอยากจะออกแบบเองก็ไม่ว่ากัน ขอเพียงให้ฟอนต์นั้นมีลักษณะเดียวกันคือ มีขนาด 8×20 จุด

ในครั้งแรกที่ CU Writer ปรากฏโฉมออกมาบนจอภาพสีเดียวหรือจอเขียว ตัวอักษรขนาดนี้แสดงผลได้ 17 บรรทัดที่ความละเอียดของจอภาพ 720×348 จุด นับว่ามีความเหมาะสมกับสายตามาก ไม่เล็กไม่ใหญ่เกินไป และอาจเป็นจุดหนึ่งที่ทำให้มีการพัฒนารูปแบบของตัวอักษรนี้เป็นตัวอย่างในงานด้านกราฟิกภาษาไทยต่อมา แต่เมื่อถึงยุคของจอภาพแบบวีจีเอแพร่หลายมากขึ้น ความสามารถทางด้านกราฟิกมีความละเอียดสูงขึ้น และได้เข้ามาแทนที่จอเขียว CU Writer ก็ยังทำหน้าที่ต่อไป รูปแบบของตัวหนังสือไม่ได้เปลี่ยนแปลงตามความละเอียดของจอภาพ แต่ก็ทำให้การแสดงผลในหนึ่งหน้าจอภาพขนาด 640×480 จุด ได้จำนวนบรรทัดมากขึ้นถึง 23 บรรทัด เป็นธรรมดาย่อมทำให้ตัวหนังสือบนจอภาพมีขนาดเล็กลง ทำให้ต้องปวดตากันมากขึ้น ก็ไม่ได้ชักชวนให้มาประท้วงกัน ใครชอบจอภาพแบบไหนก็เลือกเอาเอง แต่ผู้ที่กำลังศึกษาอย่างเราต้องใช้งานบนจอภาพและความละเอียดได้ทุกชนิด ไม่ได้เลือกที่รักมักที่ชังว่าจอนั้นใช้ได้ จอนี้ใช้ไม่ได้

ฟอนต์

เดิมทีเราใช้งานฟอนต์ขนาด 8×14 จุด ให้แสดงผลในรูปแบบธรรมดาเพียงอย่างเดียว คราวนี้ได้เพิ่มลูกเล่นพิเศษเข้ามาด้วย เช่น ตัวเอียง ตัวหนา และตัวขยาย เราก็เอาฟอนต์ของ CU Writer มาแนะนำกันตรงนี้เสียก่อน นอกจากขนาดฟอนต์ 8×20 จุดแล้ว ยังมีการจัดฟอนต์ไว้จำนวน 256 ตัวอักษร ใช้รหัสภาษาไทย สมอ. แต่ละตัวอักษรใช้ข้อมูล 20 ไบต์ (1 ไบต์เก็บข้อมูลได้ 8 จุดอยู่แล้ว 1 ไบต์มี 8 บิต) และเก็บเรียงกันจากบนลงล่างของแต่ละตัวอักษร ดังนั้นเราจะเห็นว่าทั้งไฟล์ NORMAL.FON และ ITALIC.FON มีขนาด 5,120 ไบต์ (20×256) การใช้รหัสควบคุมตัวอักษรพิเศษของ CU Writer นั้นเริ่มตั้งแต่แอสกี 2 ถึง 31 (ไม่ได้ใช้ทั้งหมด) เราจะขอเลือกมาใช้บางส่วนเท่านั้น เช่น แอสกี 2 เป็นรหัสควบคุมตัวหนา แอสกี 5 เป็นรหัสควบคุมตัวขยาย และแอสกี 23 เป็นรหัสควบคุมตัวเอียง เวลาที่เราจะแสดงผลแต่ละตัวอักษรเราจะต้องตรวจสอบรหัสควบคุมเสียก่อน โดยให้ความสำคัญจากตัวเอียง ตัวหนา แล้วถึงตัวขยาย ถึงตอนนี้ต้องพลิกไปดูโปรแกรมแล้วจะเข้าใจมากขึ้น

รูปแบบตัวอักษรธรรมดา (NORMAL) และตัวเอียง (ITALIC) เหมือนกันในโครงสร้าง ต่างกันที่รูปร่างหน้าตา ถ้าท่านผู้อ่านท่านใดอยากทำให้มีหลายฟอนต์ อย่างเช่น PC Word ของราชวิถีก็ได้ เพียงแต่สร้างฟอนต์เพิ่มขึ้นมาใหม่แล้วหาอักษรควบคุม (ASCII) เอาเอง ในที่นี้เราใช้ 2 ฟอนต์ก็เพียงพอแก่การศึกษาแล้ว แต่เราจะมุ่งมาสู่การทำตัวอักษรพิเศษ ตัวหนา และตัวขยาย ส่วนตัวขีดเส้นใต้ ตัวห้อย ตัวยกขึ้น หรืออื่น ๆ ยกไว้เป็นหน้าที่ของท่านผู้อ่านก็แล้วกัน ผู้เขียนขอบอกไว้ล่วงหน้าว่าง่ายนิดเดียว ถ้าเข้าใจในหลักการที่จะอธิบายต่อไปนี้

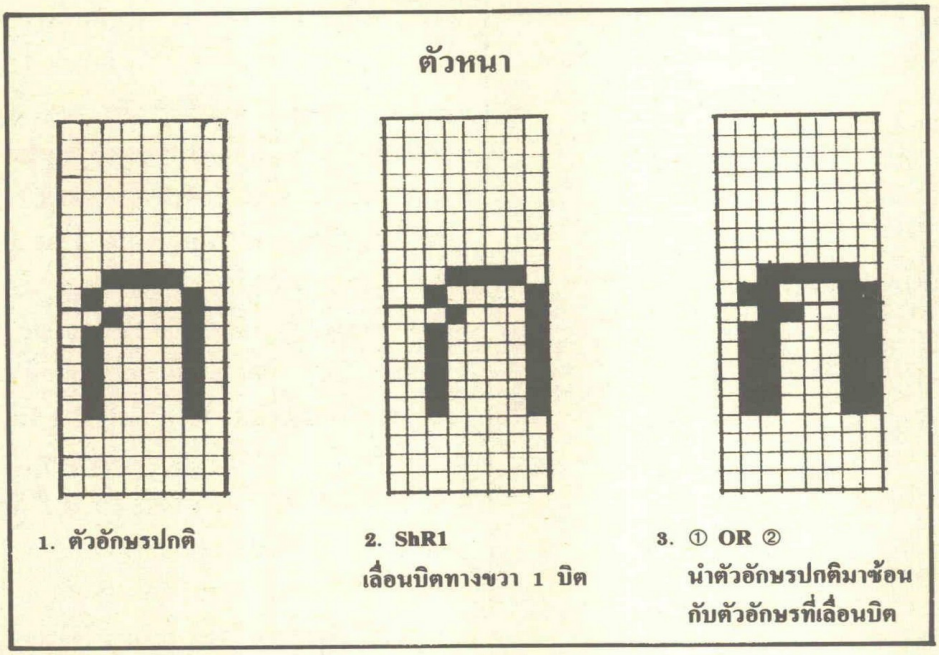
เมื่อเราได้ฟอนต์ตัวใดตัวหนึ่งแล้ว เราก็หาตัวอักษรพิเศษต่อไป ตัวอักษรพิเศษตัวนี้ก็คือ ตัวหนา หลักการก็มีไม่มาก เพียงแต่เรานำเอาแต่ละไบต์ของตัวอักษรมาเลื่อนบิตไปทางขวามือ 1 บิต และนำไป OR แบบบิตกับไบต์เดิมอีกครั้ง ทำเช่นนี้จนครบทุกไบต์เท่านั้นเอง เห็นไหมว่าง่ายจริง ๆ ลองดูตัวอย่างนี้

```
if (Bold) One__Image = (One__Image
        >> 1); หรือ
if Bold then One__Bitmap [Count]:=
        One__Bitmap [Count] or
    (One__Bitmap [Count] shr 1);
```

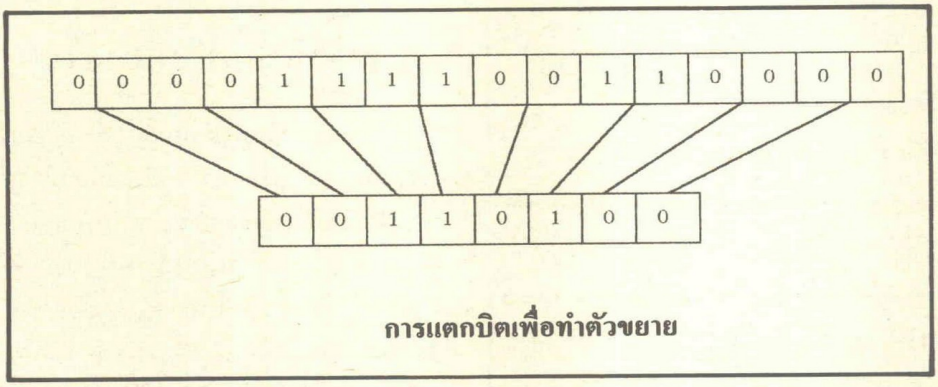
สำหรับการทำตัวหนา ถ้าดูโปรแกรมแล้วยังไม่เข้าใจ ก็ดูรูปภาพประกอบไปด้วยดังรูปที่ 1 ซึ่งเป็นรูปแบบการทำตัวอักษรจากตัวปกติให้เป็นตัวหนา โดยนำเอาตัวอักษรปกติมาทำการเลื่อนบิตไปทางขวา 1 บิต (Sh R 1) และก็นำมาทำการ OR กันอีกครั้งหนึ่ง (1 OR 2)

ตัวอักษรพิเศษอีกชนิดหนึ่งคือตัวขยาย ขั้นตอนนี้ต้องผ่านการเลือกฟอนต์และทำตัวหนามาแล้ว หลักการก็ง่าย ๆ อีกนั่นแหละ (แต่ยุ่งยากนิดหน่อย) เริ่มด้วยจากการนำเอาแต่ละไบต์ของตัวอักษรมาทำการขยายให้เป็นเวิร์ด (จาก 8 บิตเป็น 16 บิต) ดังรูปที่ 2 เราใช้เทคนิคการเลื่อนบิตอีกเช่นเคย ความจริงแล้วบางคนมีความเข้าใจเรื่องการคำนวณมากกว่าการเลื่อนบิต เช่น การคูณด้วย 2 เท่ากับการเลื่อนบิตไปทางซ้าย 1 บิต หรือการหารด้วย 2 เท่ากับการเลื่อนบิตไปทางขวา 1 บิต เนื่องจากการคำนวณทางคณิตศาสตร์ต้องมีการจัดการเกี่ยวกับตัวแปรเสียก่อน เช่น การจัดสแต็ก การใช้ ALU ทำให้วิธีการคำนวณทำงานช้ากว่าการเลื่อนบิต

ถึงตรงนี้เรามาดูการเลื่อนบิตของการทำตัวขยายแทนการคูณด้วย 4 ตัวแปร ReNum เป็นตัวแปรชนิด Word หรือ



รูปที่ 1 แสดงวิธีการทำตัวหนา



รูปที่ 2 การแตกบิตจาก 8 บิต เป็น 16 บิต

unsigned int มีค่าเริ่มต้นที่ 0003h หรือ 00000000 00000011 ในรูปของเลขฐานสอง ที่เราให้ค่าเริ่มต้นเป็น 3 เพราะง่ายต่อการเปลี่ยนบิต 1 จำนวน 1 บิต เป็นบิต 1 จำนวน 2 บิต เราใช้ ReNum สำหรับเปรียบเทียบค่าบิตที่เป็นจริง (เท่ากับ 1) เมื่อเทียบบิตแล้วจะคูณด้วย 4 หรือเลื่อนบิตไปทางซ้าย 2 บิต เทียบบิตไปเรื่อย ๆ จนครบทั้ง 8 บิต เราก็จะได้ค่าที่ตัวแปร DWNum ซึ่งเป็นตัวแปรชนิด Word หรือ unsigned int เช่นกัน แต่ทำหน้าที่เก็บค่าที่เปรียบเทียบแล้ว ถึงตอนนี้ไม่ใช่ว่าเราจะเอาค่าของ DWNum ไปใช้ได้เลย เพราะในความจริงตัวแปร Word หรือ unsigned int เมื่อมีการจัดเก็บ ก็จะเก็บเป็นไบต์จำนวน 2 ไบต์ โดยเอาไบต์ต่ำเข้าไปเก็บก่อนแล้วตามด้วยไบต์สูง (บรรดานักแกะโปรแกรมทั้งหลายที่นิยม Dump ไฟล์หรือแผ่นดิสค์มาดู เวลาอ่านค่าจึงต้องสลับไบต์ที่อ่านได้เสมอ) ดังนั้นเราจึงต้องมีการสลับไบต์ DWNum เสียก่อน เพราะการกระทำดังกล่าวเป็นการกระทำกับบิตโดยตรง ไม่ใช่กระทำในรูปของตัวแปร สำหรับภาษาปาสคาลขอยกตัวอย่างฟังก์ชัน Swap ที่ทำหน้าที่สลับไบต์สูงกับไบต์ต่ำมาใช้ แต่ในภาษาซีเราจะใช้การเลื่อนบิตไปทางซ้าย 8 บิต เลื่อนบิตไปทางขวา 8 บิต แล้วมา OR กันแบบบิต ก็จะได้ผลลัพธ์เช่นเดียวกัน

สำหรับท่านที่ยังไม่เข้าใจวิธีการทำตัวขยาย ก็ดูรูปที่ 3 ประกอบไปด้วย ก็จะทำให้ผู้อ่านได้เข้าใจวิธีการทำตัวขยายมากขึ้น

ดูตัวอย่างไปแล้วคงไม่ยากเกินไปนะครับ สำหรับการปฏิบัติงานบนบิต (เป็นภาษาที่นิยมใช้กันอยู่ก็เลยขอใช้ดูมั่งเพื่อความทันสมัย Operation on bit) เราได้เริ่มต้นจากการใช้ฟอนต์ของ CU Writer มาทำตัวธรรมดา ตัวเอียง ตัวหนา และตัวขยาย มาแล้ว

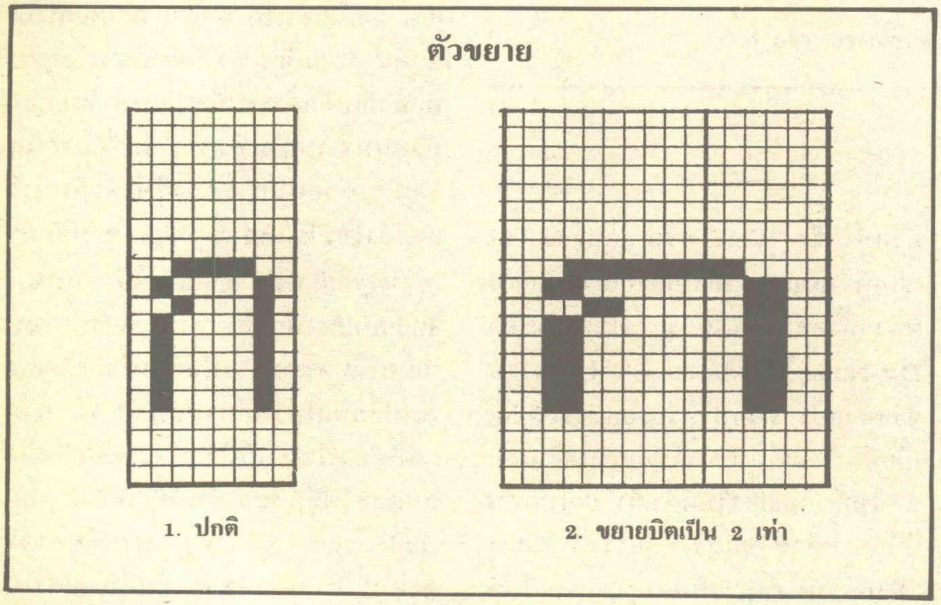
ผู้เขียนก็มีข้อแนะนำหรือจะเรียกว่าเป็นการบ้านสำหรับท่านผู้อ่านก็ได้ เพื่อฝึกฝนวิทยายุทธ์เพิ่มเติม ทดลองทำดู เช่น ตัวขีดเส้นใต้ 1 เส้น ตัวขีดเส้นใต้ 2 เส้น ตัวยกขึ้น ตัวห้อย เมื่อบรรลุขั้นนี้แล้วก็เหลือการรับข้อมูลภาษาไทยและอังกฤษจากแป้นพิมพ์ ใครมีเทคนิคใหม่ ๆ ช่วยกันถ่ายทอดให้ได้รู้กันบ้าง และถ้าใครไม่ไหวจริง ๆ ก็ต้องบอกมา เพราะผู้เขียนตั้งใจจะจบการใช้งานภาษาไทยแบบกราฟิกเพียงเท่านี้ สำหรับภาษาซีเมื่อได้ทดสอบจนเป็นที่แน่ใจแล้ว ใครจะเปลี่ยนเป็นไฟล์นามสกุล .H ก็ได้ แต่ก็ต้องนำไปไว้ที่สับไดเรคทอรีของอินคลูดไฟล์ด้วย แล้วเปลี่ยนเป็น #include <cw_font.h> ในไฟล์ของเราด้วยนะ ถ้าคิดจะใช้อีกนานก็ลิงค์เข้าไปในไลบรารีเสียเลย

```
procedure WidthChar(Space:integer;G__Col, G__Row:word;Count,One__Byte:byte);
var XCount:byte;
          ReNum,DWnum:word;
begin
ReNum :=3;
DWNum :=0;
for XCount:=0 to 7 do
  begin
    if (One__Byte and 1)=1 then DWNum:=DWNum+ReNum;
    One__Byte:=One__Byte shr 1;
    ReNum:=ReNum sh1 2;
    end;
  DWNum:=swap(DWNum);
  memw[$A000:Off__Line[G__Row+Count]+G__Col+(Space*2)]:=DWNum;
end;
```

หรือ

```
void WidthChar(char Space,byte Count,byte One__Byte)
{
  byte XCount;
  unsigned int ReNum=3,DWNum=0;
  for (XCount=0;XCount<8;XCount++)
  {
    if (One__Byte & 1) DWNum=DWNum+ReNum;
    One__Byte >>= 1;
    ReNum <<= 2;
  }
  ReNum=DWNum >> 8;
  DWNum=(DWNum << 8) : ReNum;
poke(0xA000,Off__Line[G__Row+Count]+G__Col+(Space*2),DWNum);
```

ลิสติงที่ 1 โพรซีเดอร์เพื่อกำหนดตัวขยายของภาษาซีและปาสคาล



รูปที่ 3 แสดงวิธีการทำตัวขยาย

เรายังมีงานด้านกราฟิกอีกหลายอย่างที่ต้องคุยกัน ที่แน่ ๆ ผู้เขียนยังไม่ทันจะเริ่มต้น Super VGA เลย ก็ต้องมาเจอกับวีจีเอการ์ดรุ่นใหม่ ๆ เช่น บน ISA Bus ของ Trident รุ่น 8900CL 32 บิต หรือ Accelerator ของ TSeng Lab และบน VESA Local Bus 32 บิต ทั้งแบบธรรมดาและแบบ Accelerator ปัญหาของผู้เขียนก็อยู่ที่การหาอุปกรณ์และตำรับตำรา ไม่รู้ว่าจะมีโอกาสได้ศึกษาหรือไม่ และขอขอบคุณท่านผู้อ่านที่ไม่ประสงค์ให้ออกนามได้นำเอาการ์ด Trident รุ่น 8900CL 32 บิตมาแสดงให้ดู และได้เห็นถึงปัญหาสำหรับโปรแกรมด้านกราฟิกบางโปรแกรมที่ทำงานได้ และบางโปรแกรมมีปัญหาเล็กน้อย

ภาพกราฟิกสีเป็นอย่างไร

ภาพกราฟิกดูจะเป็นเรื่องที่สามารถบ่งบอกถึงความสามารถของโปรแกรมและประสิทธิภาพของเครื่องคอมพิวเตอร์ ไม่ว่าจะเป็นความละเอียดของภาพ จำนวนสี ความเร็วในการแสดงผล ทั้งโปรแกรมบนดอสและวินโดวส์ แต่สิ่งหนึ่งที่ภาพกราฟิกจะต้องมีก็คือ ลักษณะภาพ (Format) หรือที่เราเข้าใจกันทั่ว ๆ ไปว่านามสกุล เช่น TIF, BMP, PCX, GIF หรือ PIC ไฟล์ภาพเหล่านี้โดยปกติแล้วจะประกอบไปด้วยเฮดเดอร์ (Header) ข้อมูลรูปภาพ และแพเลตต์ (Palette) เฮดเดอร์จะเป็นตัวชี้ลักษณะของรูปภาพนั้นว่ามีขนาดกว้างยาวเท่าใด เป็นฟอร์แมตชนิดใด มีจำนวนสีเท่าไร เป็นต้น ส่วนข้อมูลรูปภาพจะเป็นข้อมูลแบบเข้ารหัสหรือบีบข้อมูล ในการบีบข้อมูลมีความแตกต่างกันในแต่ละฟอร์แมต มีทั้งการบีบข้อมูลแบบนับซ้ำ (Run

Length Limit) แบบนับความถี่เชิงบิต (แบบ LZW = Lempel and Ziv (Welch 1984)) บางฟอร์แมตก็จะมีการจัดเก็บรูปภาพโดยตรง ไม่มีการบีบข้อมูลเลย สำหรับแพเลตต์นั้นเป็นการกำหนดค่าสี พอจะอธิบายสั้น ๆ ได้คือ ในการเก็บภาพไว้แต่ละภาพ ไม่สามารถเก็บเป็นค่าสีเป็นมาตรฐานได้ เช่น ภาพนกอินทรีทอง ต้องการให้สีที่ปากเป็นสีเหลืองธรรมชาติจริง ๆ แต่ค่าสีเหลืองที่มีอยู่ไม่ตรงตามความต้องการ เราจึงสามารถเซตแพเลตต์ได้ ค่าของสีแต่ละแม่สีมีอยู่ 64 (0-63) ระดับเท่านั้น จะเห็นได้ว่าสีเหลืองที่เราต้องการ สามารถกำหนดได้จากสีแดงผสมกับสีเขียวถึง 63×63 ระดับทีเดียว (ค่า 0 ไม่ได้ทำให้เกิดสีผสม เราจึงไม่นำมาคิดด้วย) ก็เพียงพอกับสายตาและความต้องการของมนุษย์ขณะนี้แล้ว (ในอนาคตค่อยดูกันอีกที)

ถึงตรงนี้ถ้าท่านผู้อ่านที่ชอบเล่นเกม ผู้เขียนขอแทรกไว้เพื่อเป็นไอเดียสักนิด เกมที่มีแบคกราวนด์ เช่น ท้องฟ้า ภาพภูเขา หรืออาคารต่าง ๆ ที่เริ่มจากมืดแล้วค่อย ๆ สว่างขึ้นมา ผู้เขียนเกมไม่ได้สร้างภาพใหม่ขึ้นมาเลย แต่เป็นการใช้เทคนิคเซตแพเลตต์ของสีท้องฟ้าขึ้นใหม่แทนเท่านั้นเอง ถ้าไม่เชื่อก็ลองทำดู ถ้าทำไม่ได้ก็รอหน่อย เมื่อเราสามารถตั้งค่าของสีได้ เราก็ต้องเก็บค่าของสีได้ด้วย ดังนั้นแพเลตต์จึงถูกเก็บมาพร้อมกับไฟล์ของรูปภาพ บางฟอร์แมตเก็บแพเลตต์ไว้ที่ส่วนเฮดเดอร์ หรือต่อจากเฮดเดอร์ หรือท้ายไฟล์ เป็นต้น แต่ตอนนี้เราต้องการแสดงภาพที่ได้จากการศึกษาแพลนสีทั้ง 4 แพลนมาแล้ว ผู้เขียนจึงเลือกเอาฟอร์แมต PCX มาเป็นตัวอย่างให้ดู

เล็ก ๆ น้อย ๆ PCX Header

PCX เป็นรูปแบบไฟล์กราฟิกของบริษัท Zsoft มีใช้ในโปรแกรมพีซีเพนต์บรัชและ Publisher's Paintbrush ใครจะเอาไปใช้อีกก็ได้ แต่ต้องมีการจัดเก็บรูปภาพให้ครบทั้งสามส่วนจากที่กล่าวมาแล้ว อีกทั้งยังมีการบีบข้อมูลแบบ RLL ด้วย ถ้าจะไม่อธิบายลักษณะไฟล์ของฟอร์แมต PCX เลยก็เกรงว่าเมื่อดูตัวอย่างโปรแกรมแล้วจะงงมากขึ้น แต่ขออธิบายเพียงสั้น ๆ ก็คงไม่ว่ากันนะ และเราจะใช้ตัวอย่างภาพที่มีความละเอียด 640×480 จุด ที่ 16 สี

ไฟล์ PCX มีส่วนของเฮดเดอร์อยู่ 128 ไบต์ เราจะแกะเอาส่วนที่สำคัญ ๆ มาใช้งานก่อน ที่เหลือ ๆ ยกให้เป็นหน้าที่ของผู้อ่านไปศึกษาเองบ้าง จะได้สมกับโปรแกรมเมอร์มือใหม่ เมื่อแกะแล้วเราจะได้ดังนี้

- ไบต์ที่ 0 0Ah เป็นรหัสของ Zsoft PCX file
- ไบต์ที่ 3 จำนวนบิตต่อจุด
- ไบต์ที่ 4, 5 จำนวนจุดน้อยที่สุดบนแกน X
- ไบต์ที่ 6, 7 จำนวนจุดน้อยที่สุดบนแกน Y
- ไบต์ที่ 8, 9 จำนวนจุดมากที่สุดบนแกน X
- ไบต์ที่ 10, 11 จำนวนจุดมากที่สุดบนแกน Y
- ไบต์ที่ 16 ถึง 63 จำนวน 48 ไบต์ คือ แพเลตต์
- ไบต์ที่ 65 จำนวนแพลนที่ใช้
- ไบต์ที่ 66 จำนวนไบต์ต่อหนึ่งเส้นภาพ

เอาแค่นี้ก็เพียงพอแล้ว เมื่อเราพอจะมองเห็นส่วนของเฮดเดอร์แล้ว ส่วนต่อไปเป็นเรื่องที่เราสนใจเอามาก ๆ ทีเดียวคือ ข้อมูลของรูปภาพ

PCX เข้ารหัสอย่างไร

ไฟล์ PCX ในที่นี้เราต้องการข้อมูลแบบบีบข้อมูลชนิด (run length encoding) ไฟล์ PCX จะเก็บข้อมูลรูปภาพที่มีค่าเหมือนกันไว้เป็นจำนวนนับก่อน แล้วมา XOR กับ C0h จึงจะตามด้วยข้อมูลจริงอีกที ดังนั้นข้อมูลที่นับซ้ำจะมีค่ามากกว่า 63 ไม่ได้ ข้อมูลที่มากกว่า 63 จะต้องสร้างตัวนับใหม่อีกชุดหนึ่ง ยกตัวอย่างเช่น มีข้อมูล FFh จำนวน 8 ตัว เมื่อเรา XOR กับ C0h แล้วก็จะได้ค่าเป็น C8h และ FFh หรือถ้ามีข้อมูล FFh จำนวน 32 ตัว เมื่อเรา XOR กับ C0h ก็จะได้ค่าเป็น E0h และ FFh ที่เราไม่สามารถให้ค่าตัวนับมากกว่า 63 นั้น เนื่องจากตอนที่เราทำการถอดรหัส เราใช้การเทียบค่าว่าข้อมูลนั้นมากกว่า C0h หรือไม่ ถ้ามากกว่าเราถือว่าข้อมูลนี้เป็นตัวนับ (เมื่อ XOR กับ C0h ก็จะได้ค่าตัวนับ) ข้อมูลตัวต่อไปถือว่าเป็นข้อมูลรูปภาพทันที เช่น มีข้อมูล D0h จำนวน 63 ตัว เมื่อเรา XOR กับ C0h แล้วก็จะได้ค่าเป็น FFh และ D0h ถ้าข้อมูล D0h จำนวน 64 ตัว เมื่อเรา XOR กับ C0h แล้วก็จะได้ค่าเป็น 80h และ D0h เมื่อถึงตอนถอดรหัส 80h กลายเป็นข้อมูลภาพ D0h ก็กลับเป็นตัวนับ ซึ่งผิดกับความเป็นจริงมาก และที่ถูกต้องก็คือ FFh, D0h และ C1h, D0h ดูตัวอย่างตอนถอดรหัสแล้วอาจจะทำให้เข้าใจได้ดีขึ้น

PCX แพเลตต์ 16 สีอยู่ที่ไหน

เมื่อเราสามารถถอดรหัสภาพฟอร์แมต PCX ได้แล้ว เราก็ถึงจุดสำคัญสุดยอดของเรื่องแล้ว ที่ว่าสุดยอดนี้ก็เป็นเรื่องของแพลนสีที่เราได้ศึกษามาแต่ตอนต้น PCX ไฟล์เก็บเส้นภาพตามแนวนอนแต่ละเส้นไว้ 4 ชุด ชุดแรกเป็นของแพลนหมายเลข 1 และแพลนหมายเลข 2, 4, 8 ตามลำดับ แล้วก็เริ่มเส้นต่อไปอีก 4 ชุด ไปเรื่อย ๆ จนจบภาพทั้งหมด เวลาที่เราต้องการแสดงบนจอภาพ เราก็ต้องอ่านข้อมูลภาพของเส้นแรก แล้วส่งไปแสดงผลยังแพลนหมายเลข 1 ก่อน เมื่ออ่านข้อมูลต่อไปก็ยังคงเป็นเส้นแรก แต่เป็นของแพลนหมายเลข 2 ดังนั้นเราจะเลื่อนเส้นภาพการแสดงผลในแนวนอนแต่ละเส้นได้ต้องอ่านข้อมูลให้ครบ 4 ชุดก่อนเสมอ แล้วเราก็จะได้การผสมสีของภาพครบทั้ง 16 สีดังปรารถนา

แต่ยังไม่จบพระเอกของเรื่องยังไม่ได้ของวิเศษไปเลย แพเลตต์ไงล่ะ เกือบลืมของสำคัญเสียแล้ว เมื่อเราได้ภาพแล้วลองเปรียบเทียบดูว่าก่อนการเซตแพเลตต์กับหลังจากเซตแพเลตต์แล้วสีสันที่ได้จะต่างกันอย่างไร แล้วเจ้าแพเลตต์ที่ว่านี้มันอยู่ที่ไหน ที่ผ่านมารูปภาพ 16 สีอยู่ที่เฮดเดอร์จำนวน 48 ไบต์ แต่รูปภาพ 256 สี อยู่ที่ท้ายไฟล์จำนวน 768 ไบต์ ...เล่นเอาท่านผู้อ่านงงอีกแล้วซิ ผู้เขียนไม่ได้ทำให้งง แต่คนที่เขาสร้างฟอร์แมต PCX เขามีเหตุผลของเขา เรามีหน้าที่ศึกษาก็ต้องศึกษากันต่อไป ที่จริงแล้วเป็นเรื่องง่าย ๆ ถ้ายังจำได้ว่าเรามีแม่สีแสงอยู่ 3 สี คือ สีแดง สีเขียว และสีน้ำเงิน (RGB) การเซตแพเลตต์ไม่ได้กำหนดให้เซตทีเดียวครบทั้ง 3 ค่าสี แต่เพื่อความสะดวกและง่ายต่อการจัดเก็บเราจึงนิยมเซตแพเลตต์ทีเดียวทั้ง 3 ค่าสี เอาล่ะเมื่อเซต 3 ค่าสีทั้ง 16 สี ก็จะได้เท่ากับ 3×16 = 48 ไบต์ พอดี และก็ไม่ต่างกับ 256 สีเลย 3×256 = 768 ไบต์เช่นกัน

การเซตแพเลตต์นั้นทำได้หลายวิธี จะใช้การผ่านค่าทางพอร์ต หรือการเรียกอินเตอร์รัพต์ก็ได้ แล้วแต่ความชอบหรือถนัดของแต่ละคน แต่ต้องไม่ลืมว่าทุกค่าของแพเลตต์ PCX ฟอร์แมตจะนำไปคูณด้วย 4 เสมอ เมื่อต้องการค่าจริง ๆ จะต้องนำไปหารด้วย 4 หรือเลื่อนบิตไปทางขวามือ 2 บิตก่อน ท่านผู้อ่านท่านใดที่กำลังคิดว่าจะเลือกเรียนภาษาปาสคาลหรือภาษาซีอย่างไหนจะดีกว่า ก็ลองเอาตัวอย่างการแสดงภาพ PCX ที่ผู้เขียนเสนอไว้ในตอนนี้ไปลองรันดู แล้วจะเห็นถึงขนาดของไฟล์และความเร็วที่แตกต่างกันอย่างมาก

```
while not(eof(PCX__Byte)) and (RowNum<MaxRow) and (RowNum<Hight) do
begin
  read(PCX__Byte,Data);
  if Data>$C0 then
    begin
      Count:=Data xor $C0;
      read(PCX__Byte,Data);
      for XCount:=1 to Count do ColorPCX(Data,Plane);
    end
else
  ColorPCX(Data,Plane);
end;

หรือ

while (!feof(Fp) && RowNum<MaxRow && RowNum<Hight)
{
  fread(&Data,1,1,Fp);
  if (Data>0x00C0)
  {
    Count=Data&0x00C0;
    fread(&Data,1,1,Fp);
    for (Xcount=0;Xcount<Count;Xcount++) ColorPCX(Data,&RowNum,Width);
  }
  else
    ColorPCX(Data,&RowNum,Width);
}
```

ลิสติงที่ 2 โปรแกรมย่อยสำหรับกำหนด .PCX แพเลตต์สำหรับภาษาซีและปาสคาล

ก่อนจาก

ผู้เขียนฝากภาพ TRAIN.PCX มาให้เป็นตัวอย่างด้วย ตามหาในซอฟต์เมล์ได้เลย

ผู้เขียนขอค้างไว้อีกเรื่องก็คือ เรื่องการอ่านข้อมูลจากหน่วยความจำ ผู้เขียนจะเอาภาพเคลื่อนไหวมาเป็นตัวอย่างให้สมกับที่ต้องขอผลัดไป แต่คราวหน้าเป็นเรื่องของจอวีจีเอล้วน ๆ นะครับ...

พบกันใหม่ฉบับหน้า สวัสดีครับ